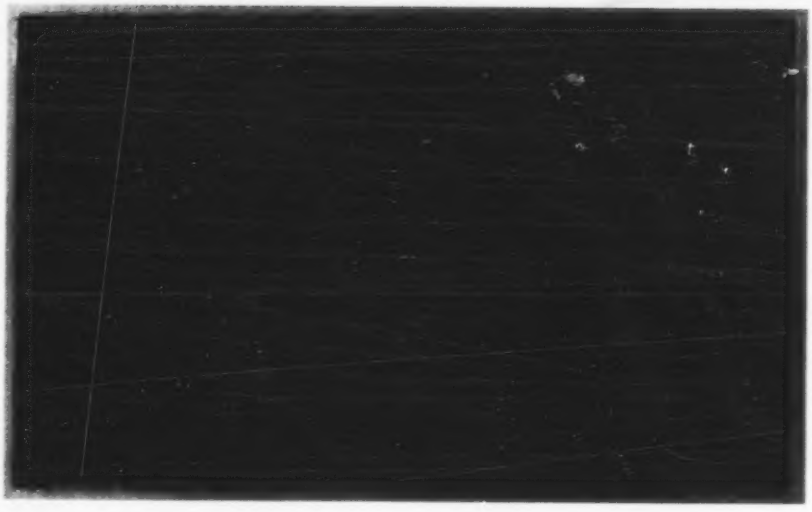
CENTRE DE RECHERCHE SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE



CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-561325
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

91-06352/

CRT-774

The Dynamic Predicate Stashing Copy Problem

by

Brunilde Sansó [1,2,3]
François Soumis [2,3]

[1] Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Québec H3C 3J7, tel.: (514) 343–2458, Email bruni@crt.umontreal.ca.
[2] GERAD, École des Hautes Études Commerciales, 5255, avenue Decelles, Montréal, Québec H3T 1V6
[3] Ecole Polytechnique de Montréal, C.P. 6079, succursale A, Montréal, Québec H3C 3A7

Centre de recherche sur les transports — Publication #774

Juillet 1991

Dépôt légal – 3ème trimestre 1991

RÉSUMÉ

Un problème important qui a été adressé dernièrement dans le design de réseaux distribués d'ordinateurs est celui d'augmenter la disponibilité de l'information. Récemment, une technique appelée "stashing" a été proposée. "Stasher" un fichier veut dire placer les fichiers dans plusieurs noeuds du réseau pour augmenter sa disponibilité.

Les copies "stashées" dans le réseau sont appelées "quasi-copies" car elles ne sont pas nécessairement identiques aux copies gardées par le serveur, cependant elles réunissent des restrictions de consistance appelées "prédicats" et établies par les usagers. Les prédicats peuvent être traduits en terme de l'âge de la quasi-copie.

Nous adressons le problème de minimiser la propagation de copies, à "stasher" dans un réseau étant donné que les contraintes d'âge des quasi-copies sont satisfaites. Dans la version du problème que nous traitons l'usager peut recevoir une copie plus jeune qu'il ne l'a demandé.

Mots-clés: "stashing", réseaux d'ordinateurs, réseaux distribués, prédicats, optimisation, gestion d'information.

ABSTRACT

One important problem that is being addressed in the design of distributed networks is how to increase the availability of information. A technique called "stashing" has been proposed for this purpose. Stashing a file means placing a copy of that file in several nodes of the network to increase its availability.

On the other hand, a group of researchers in the field have combined the idea of stashing with "quasi-copies" or imperfect replicas of a file and have suggested an application where the user could specify how different a copy can be by means of a predicate requirement.

We address the problem of minimizing the stashing copies propagation cost subject to time predicate restrictions imposed by the user. In the version of the problem we tackle, the user may receive a copy with a predicate better than requested. We call this problem the Dynamic Predicate Stashing Copy Problem (DPSCP) and propose a formulation to solve it.

Resolution approaches, numerical results and properties of the problem are also discussed.

Keywords: Stashing, hoarding, distributed systems, predicates, optimization, information resource management.

1 Introduction

There are several definitions that can be found regarding the term "distributed computer system". The most colorful is attributed to Leslie Lamport and cited by several authors: "a distributed system is one that stops you from getting any work done when a machine you never heard of crashes". A more formal definition can be found in Mullender's book (see Mullender (1989)) where a distributed system is considered "a set of processing elements interconnected by hardware, that do not share physical memory". The desirability of distributed computer systems rather than centralized systems has been a matter of discussion among the experts in the field; however, whatever is the outcome of the discussion, we are definitely evolving toward larger distributed systems. One point usually claimed in favor of distributed systems is that they allow, at the same time, autonomy and economy of resources. As distributed systems become larger, the problem of how to manage resources and replicated information so that autonomy and efficiency is maintained becomes more important. One common technique to improve the overall performance of the system is to use redundancy (see for instance, Walker (1983) and Chapters 13 - 15 of Mullender(1989)). Another technique is called *stashing*, which was formally introduced by Birrel(1988).

Stashing a file has been defined as the process of placing a copy of that file in several nodes of the network in order to increase file availability. To fully understand the notion of stashing, we believe it is necessary to introduce the notion of a server. A *server* is generally defined as a dedicated machine of the system that provides a specific service (i.e. sending mail, file storage, etc.). Very often, in distributed computer sytems, if the server is not available and there is no server redundancy, the customer will not be able to accomplish the tasks usually required from the unavailable server. Therefore, in most cases, the user will be more glad to have access to old information that had been *stashed* to some local devices than not working at all on his particular application.

The idea that old information could be useful to the users was considered by Alonso and Cova (1990). They combined the notion of stashing with the concept of *quasi-copies*, or imperfect replicas of a file, to increase file availability. A quasi-copy of a particular file was defined as a copy that is not exactly the original but meets some minimum consistency requirements. Those requirements are called *predicates*.

Let us suppose that the computer network is divided into server and customer machines. The server keeps original updated files and send copies to the customers. If the copy on which the customer is working is not identical to the one kept by the server, the customer is then working with a quasi-copy. The degree of consistency between the quasi-copy and the original is pre-specified by the customer by means of a predicate. A predicate can also be translated in terms of time. For instance, the customer may ask to receive a new copy every three minutes. We call this type of predicate "static" since the quasi-copy kept by the customer machine is going to be replaced when it is exactly three minutes old.

Another type of predicate could be an upper bound on the age of a quasi-copy. For instance, in the example given above, the customer could specify the quasi-copy to be *at most* three minutes old. We call this type of predicate "dynamic" since the quasi-copy can be replaced at different ages: when it is one minute, two minute or three minute old.

The problem tackled in this paper, that we refer to as the **Dynamic Predicate Stashing Copy Problem (DPSCP)**, is the scheduling of the stashing of customer machines in a distributed network given that transmission costs are minimized and dynamic predicate requirements stated by the customers are met. We mention that the problem in which the predicates are static (i.e. the customer never receives a copy more consistent than required) has been solved by Cova (1990) in polynomial time. To our knowledge, the DPSCP, which was presented as an open problem in Cova's dissertation, has not been formally introduced in the literature.

The objective of this paper is to define the DPSCP, present a formulation, a resolution approach and some extensions of the problem that will be presented in forthcoming papers.

The paper is organized as follows. In section 2, a detailed description of the problem is given. In section 3, a review on similar problems treated in the literature is presented. The problem is formulated in section 4 and, in section 5, some numerical examples are shown. Section 6 contains some conclusions and recommendations for further work.

2 The Problem

In this section, we formalize the problem and explain its main characteristics. First of all let us suppose $G(N, A)$ is the underlying graph for the computer network. N, the set of nodes, represents the computers and A, the set of arcs, represents the links between computers. For simplifying purposes, we suppose that there is just one node where an exact copy of a file can be found, this node will be indistinctly called *server* or *root* to make the analogy with standard graph jargon. The problem could, however, be easily generalized to several files and servers.

The propagation of a file in the network is made possible by duplicating that file on some nodes, referred to here as *intermediate*. On the other hand, the *customer* machines, that have to be stashed with copies of the file are represented by a set of nodes called *destination nodes*, denoted by D. Destination nodes may be intermediate nodes as well.

Let us now discuss the temporal aspect of the problem. The file copies are going to be transmited in the network at particular instants of time. Therefore, the time space must be discretized in units of time. The discretization unit should be chosen according to how often the destination nodes have to be stashed. In fact, the discretization unit should be the maximum common divisor of all the different time predicate requirements in the network. For instance, if the requirements are all multiples of four, 4 should be the discretization unit. The set of all discretized units of time is denoted by T.

The costs associated with this problem are the costs of the trees connecting the root to the destination nodes for every period of time. The problem can then be seen as the problem of finding the best trees to update all the destination nodes in a space-time graph given that the predicate requirements are met. A formal formulation is presented in the next section.

3 The Model

First of all, let us define the decision variables of the problem as the variables that determine whether a particular arc will be used at a particular time. If we let such

variables be denoted by x's, then x_{ij}^t will take the value 1 if arc (i,j) is used at time t and the value zero otherwise.

We now discuss the predicate requirements in mathematical terms. As we explained earlier, a predicate can be translated into a time requirement on how old a stashed copy can be. For every destination node $j \in D$ the requirement will be represented by an interval Δt_j that is the maximum time node j can keep the same copy of the file. For instance, if node j specifies being stashed at most every 5 minutes, Δt_j will be 5 minutes.

For the case in which graph G is directed, the DPSCP can be formulated as follows; the case for undirected networks will be discussed later.

DPSCP:

$$\min \sum_{t \in T} \sum_{(i,j) \in A} c_{ij} x_{ij}^t \tag{1}$$

s.t.

$$\sum_{i \in N} x_{ij}^t \geq x_{jk}^t, \quad \forall j \in N - r, \ \forall k \in N - r, \ \forall t \in T, \tag{2}$$

$$\sum_{i \in N} \sum_{s \in [t - \Delta t_j, t-1]} x_{ij}^s \geq 1, \quad \forall j \in D, \ \forall t \in T, \ t > \Delta t_j, \tag{3}$$

$$x_{ij}^t \ \ binary. \tag{4}$$

The objective function in the model indicates that the messages have to be transmitted at minimum cost. Note that the cost associated with an arc (i,j) is supposed to be independent of time t at which this arc must be used. However, if that was not the case, a triple index cost should be used, i.e. c_{ij}^t.

The message propagation constraint (2) guarantees that if node k receives a copy from node j ($j \neq r$) at a particular time t, node j must have received a copy at the same time. In other words, this constraint forces the copy to be propagated in a tree at every time considered. The stashing constraint(3) states that every destination node receives at least one fresh copy of the file in the interval of time required by its predicate.

4 Literature Review

As mentioned in the introduction, the DPSCP was proposed as an open problem in Cova's dissertation. The idea of stashing itself is attributed to Birrell. Details on how to maintain the stashing and a discussion on the differences between stashing and caching can be found in Alonso, Barbara and Cova (1990). The integration of the notion of quasi-copies and stashing is presented by Alonso and Cova (1990). In the former paper, an actual experimental implementation of a network where stashing of some files is allowed is described. However, only the case in which the predicates stated by the users are static was considered. Another paper that describes an implementation where some sort of stashing will be done (called "hoarding") is presented by Kistler (1990).

Concerning the mathematical problem itself, we have not found any optimization problem with the same characteristics of the DPSCP. However, we review here problems that could have some similarities with ours.

Balachandran and Deshmuk (1976) proposed the problem of dynamically storing different data records in different storage devices in each period of time so that the expected cost over a planning horizon is minimized and the particular demands for some records are satisfied for all periods of time considered. The problem has in common with the DPSCP the fact that decisions have to be taken dynamically regarding the storage of information and that an optimization model was proposed. However, the authors solved a special case of the problem in which time decisions are independent from one another, thus solving an optimization problem for every period of time considered. Furthermore, the cost structure is completely different from the one of the DPSCP. In fact, the costs considered where the costs of storing, accessing or transferring information from a device to another device whereas in the DPSCP the cost considered are transmission costs, which gives rise to a tree-like cost structure.

The DPSCP may be wrongly identified with the classical vehicle routing problem with time window constraints (VRPTW) such as the one treated by Desrosiers, Soumis and Desrochers (1984). The problem consists of finding the best routes so that vehicles visit all customer nodes at minimum cost given that customers must be visited within a time interval. One may think that the predicate requirements can be modeled as time windows and that nodes to be stashed could be seen as tasks that have to be performed

within a time window. Unfortunately, time windows are static. They start and end always at the same value of time, whereas the predicates of the DPSCP are dynamic requirements: a window on the time a particular node must be stashed starts at the time determined by the preceding stashing. Moreover, in the VRPTW the costs correspond to the path lengths between an origin and a destination. If the same origin has two destinations, then the costs will be accounted twice on the common links. Not so in the DPSCP where the transmission costs from the same origin are accounted once on the common links (referred here as a tree-like cost structure).

Another type of approach that we have found in the review is the study of on-line algorithms. One definition of this type of algorithms can be found in Manasse, McGeoch and Sleator (1988): "an on-line algorithm is an algorithm that must handle a sequence of requests, satisfying each request without knowledge of future requests". Typical problems that can be solved by on-line algorithms are server, caching and migration problems (see, for instance, Karlin, Manasse, Rudolph and Sleator (1988), and Manasse, McGeoch and Sleator (1990)). Even though many typical problems related to the storage of data and management of resources in computer networks can be modeled as on-line problems, that is not the case for the DPSCP. In fact, the main characteristics of the problem are the dynamic predicates, which must be known in advance, they therefore do not correspond to a real time situation. We mention, finally, the work by Eppstein et al. (1990) in which the problem of maintaining a minimum spanning forest in a dynamic planar graph is treated. The dynamic aspects of that problem are represented by real time cost changes and deletion or addition of vertex and edges. This is different to the DPSCP where the dynamic aspect is included in the time restriction itself.

The problem found in the literature which has a structure closer to the DPSCP is a formulation of the postal relay boxes problem presented by Bouliane in his master's thesis (see Bouliane (1988),Chapter 4). The problem is that of determining the optimal location of relay boxes given that there is a weight constraint on the quantity of mail that can be carried by a mailman. The routes, candidate box locations and the amount of mail (in pounds) carried between those locations are given; we also mention that a single box can serve more than one route. The objective is to minimize the number of relay boxes. The problem was solved by using a set covering formulation (see Bouliane and Laporte (1989)). However, in Bouliane's thesis, there is the mention of a second formulation to that problem. The formulation, which was not solved, can be seen for

our comparison purposes in the following manner (the reader should be aware that this is not exactly the interpretation given by Bouliane). Let us create a graph where the root is the central post-office and there are links between this root and the candidate box locations. Depending on the location of the box on a particular route, the mailman could pass several intersections without needing to refill his box and without exceeding the weight constraint. The analogy with the stashing is the following: stashing a node is equivalent to refilling a mailbag, or, in other words, placing a relay box in a particular node of a particular route. Furthermore, the predicate requirements of the DPSCP are similar to the weight requirements of the relay box problem. In one case, they determine how long a node can be without receiving a new message, and in the other case how far a mailman can go without refilling his mailbag.

In the postal relay box problem, the tree is at most of depth two level, that is, there is the root at the first level, the possible box location nodes at the second level, and the routes deserved by every box at the third level. The costs associated with the problem are unitary, that is, the costs of choosing a particular box. The trees to be found in the DPSCP can have any depth depending on the topological structure of the problem and the transmission propagation costs are non-unitary. This makes the DPSCP a much more complex problem in terms of easiness of resolution.

5 Resolution Approach

The general DPSCP is clearly an NP-complete problem since it can be reduced to an instance of the Steiner-tree problem (see Garey and Johnson (1979)). Therefore, no polynomial time algorithms are expected to be found to solve this problem. However, small-sized problems should be solved quite efficiently by available binary programming packages.

We first present an example where the underlying graph is a directed graph. Later, we will discuss the modifications that are necessary to solve the case of non-directed underlying graphs.

5.1 A directed network example

Figure 1 portrays an example where the underlying graph of the computer network is directed. In this and other examples, the root node will be node 1; the nodes filled in black are the nodes that need to be stashed, i.e. the destination nodes. The other nodes are considered as intermediate nodes of the network. The numbers on the arcs represent the arc costs while the numbers inside the brackets, at each destination node, represent the predicate requirement, in minutes, of that particular node. The time horizon has therefore been discretized in minutes since we consider that the server makes a new copy of the file every minute. Another assumption made in the example is that at the beginning, prior to the study, every node had a fresh copy of the file.

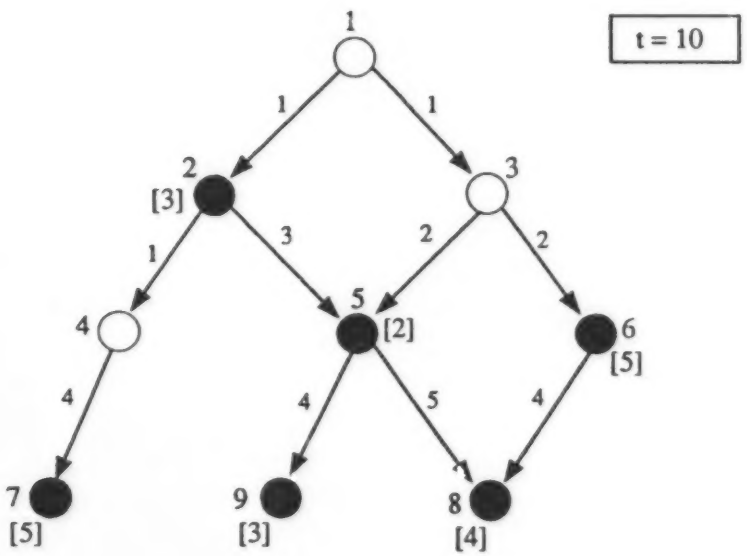


Figure 1: Directed network example

The problem was solved with the LINDO binary programming feature on a SUN4 Sparc1 workstation in 2 seconds. Results are tabulated in Table 1. In this table we show,

for every instant of time when stashing was necessary the space variables that are equal to one, the nodes that are stashed and the time at which the stashing is done. The costs of the solution and of the linear programming relaxation are given at the end.

VARIABLES	NODES STASHED	STASHING TIME
x_{13}, x_{35}	5	t=2
$x_{12}, x_{25}, x_{58}, x_{59}$	2,5,8,9	t=3
$x_{12}, x_{13}, x_{24}, x_{35}, x_{36}, x_{47}, x_{59}$	2,5,6,9	t=5
$x_{13}, x_{35}, x_{58}, x_{59}$	5,8,9	t=7
x_{12}, x_{13}, x_{35}	2,5	t=8

COST OF THE LP RELAXATION	45
COST OF THIS SOLUTION	47

Table 1: **Results for the directed network example**

After experimenting with different network examples, we found that if the number of periods considered in the study is big enough, the solution presents some kind of periodic behavior. We believe that the study of this type of behavior goes beyond the scope of this paper. We have therefore stated a conjecture that will be proven in a forthcoming work on the periodic properties of the DPSCP. Let S_j represent the state of the system defined by the age of the quasi-copy at each destination node. We can now state the conjecture.

Conjecture 1 *:The state space of the DPSCP presents some kind of periodic behavior that depends on the initial state of the system and on the predicate requirements.*

To support this conjecture, we have modified the example above so that node 6 requires to be stashed every minute, and there are just three nodes, 2, 5 and 6 to be

stashed. The example is shown in Figure 2.

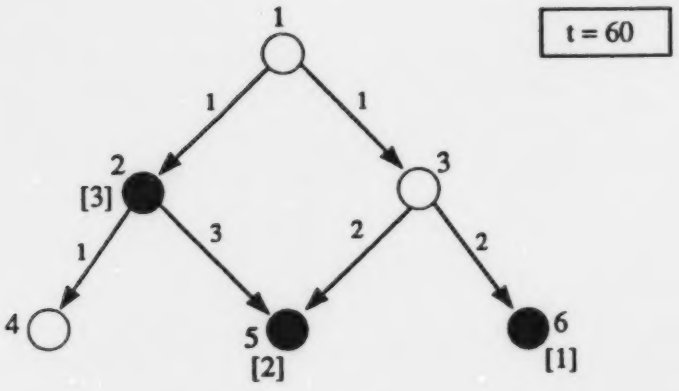


Figure 2: A network needing more frequent stashing

The problem was solved for 60 periods of time and results up to $t = 24$ are shown in Table 2.

VARIABLES	NODES STASHED	STASHING TIME
x_{13}, x_{36}	6	t=1
x_{13}, x_{35}, x_{36}	5,6	t=2
x_{12}, x_{13}, x_{36}	2,6	t=3
x_{13}, x_{35}, x_{36}	5,6	t=4
x_{13}, x_{36}	6	t=5
$x_{12}, x_{13}, x_{35}, x_{36}$	2,5,6	t=6
x_{13}, x_{36}	6	t=7
x_{13}, x_{35}, x_{36}	5,6	t=8
x_{12}, x_{13}, x_{36}	2,6	t=9
x_{13}, x_{35}, x_{36}	5,6	t=10
x_{13}, x_{36}	6	t=11
$x_{12}, x_{13}, x_{35}, x_{36}$	2,5,6	t=12
x_{13}, x_{36}	6	t=13
x_{13}, x_{35}, x_{36}	5,6	t=14
x_{12}, x_{13}, x_{36}	2,6	t=15
x_{13}, x_{35}, x_{36}	5,6	t=16
x_{13}, x_{36}	6	t=17
$x_{12}, x_{13}, x_{35}, x_{36}$	2,5,6	t=18
x_{13}, x_{36}	6	t=19
x_{13}, x_{35}, x_{36}	5,6	t=20
x_{12}, x_{13}, x_{36}	2,6	t=21
x_{13}, x_{35}, x_{36}	5,6	t=22
x_{13}, x_{36}	6	t=23
$x_{12}, x_{13}, x_{35}, x_{36}$	2,5,6	t=24

Table 2: **Results showing the periodicity of the state space**

5.2 Undirected networks

As we mentioned earlier, the formulation of the DPSCP given by (1) to (4) is valid only for directed networks. In fact, such a formulation does not guarantee that there will be no *subtours* in the solution in the case of undirected networks. From the practical point of view, it is always possible to add restrictions to the way computers communicate among themselves so that the underlying graph can be represented by a directed

network. However, one would like to be able to solve the problem for a case as general as possible. Therefore, in this subsection we tackle the problem of how to modify the given formulation so that the result would not present cycles for the general undirected network and for some mixed directed and undirected networks with particular topologies.

One way of eliminating the possibility of subtours is to add subtour elimination constraints to the formulation of the DPSCP. Firstly, the undirected links must be doubled and replaced by directed links. The original undirected network is thus replaced by a directed network with twice the number of links. Now, let S, $(S \subset N)$, be a subset of vertices such that a subtour joining those vertices can be formed. Then, the subtour elimination constraints can be stated as:

$$\sum_{i,j \in S} x_{ij}^t \leq |S| - 1 \qquad \forall t \in T, \quad S \subset N, \quad 2 \leq |S| \leq n-2. \tag{5}$$

Then, for a general undirected network,the DPSCP can be formulated as follows:

(DPSCP-g): Min (1) subject to (2) to (5).

Subtour elimination constraints are costly in terms of resolution time. In fact, for the more general case, the number of constraints grows exponentially with the number of nodes in the network. In practice, depending on the application, one may either predict or find the lengths of the likely subtours and add to the DPSCP just the constraints that correspond to those lengths. The procedure, in the second case, would be to solve the DPSCP once, detect the lenght of the cycles that are formed, and then add just the restrictions to eliminate subtours of those lengths.

To illustrate the procedure, we present, in Figure 3, an example of an undirected

network.

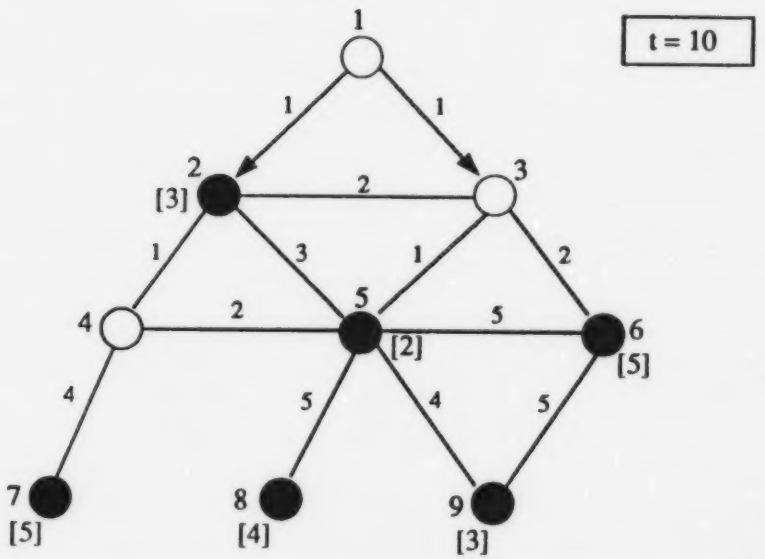


Figure 3: an undirected network example.

When examining the transmission costs given for the example, it is clear that if subtours are formed, they will be at most of length two. This means that we just need to add subtour elimination restrictions for the case in which $|S| = 2$. That is:

$$x_{ij}^t + x_{ji}^t \leq 1 \qquad \forall (i,j) \in A, \quad \forall t \in T. \tag{6}$$

In Table 3, we show the solutions for this example by solving it as a DPSCP for directed networks (without subtours elimination constraints) and as a DPSCP-g, that

is, with subtour elimination constraints. The number of periods considered, $|T|$, was set to 10.

DPSCP	DPSCP-g	STASHING TIME
-	x_{13}, x_{35}	t=1
x_{12}, x_{13}, x_{35}	-	t=2
$x_{13}, x_{35}, x_{58}, x_{59}$	$x_{12}, x_{13}, x_{35}, x_{58}, x_{59}$	t=3
$x_{13}, x_{24}, x_{35}, x_{36}, x_{42}, x_{47}, x_{59}$	$x_{12}, x_{13}, x_{24}, x_{35}, x_{36}, x_{47}, x_{59}$	t=5
x_{35}, x_{53}, x_{58}	$x_{12}, x_{13}, x_{35}, x_{58}$	t=7
$x_{12}, x_{35}, x_{53}, x_{59}$	x_{13}, x_{35}, x_{59}	t=8

Table 3: **Comparative results for the undirected network example**

It can be seen how the simple subtour of length 2 elimination constraints avoids any type of cycles in the solution. The number of such constraints is of $O(n|T|)$. Therefore, it would be interesting to find a relationship among the costs, the network structure and the type of subtour elimination constraints that are required. In the next subsection we tackle the problem of finding such a relationship for particular types of networks.

5.3 A "spiderweb" network

In this section, we present a particular type of planar network that could be seen as a generalization of some topologies used in computer networks. We define a "spiderweb" network as one made up of levels, each level containing one or more nodes. For our DPSCP application, we consider that the first level will always contain the server node. Nodes in a particular level can send messages to nodes in the same level or to nodes to the next (higher) level. Each node of a given level has at most two nodes adjacent to it that belong to the same level. Figure 4 depicts a general spiderweb network. As it can be seen, the network is portrayed as being made up of concentric rings, where the nodes are placed in the rings. The inner rings are able to communicate with the adyacent external

ring, the opposite is not allowed.

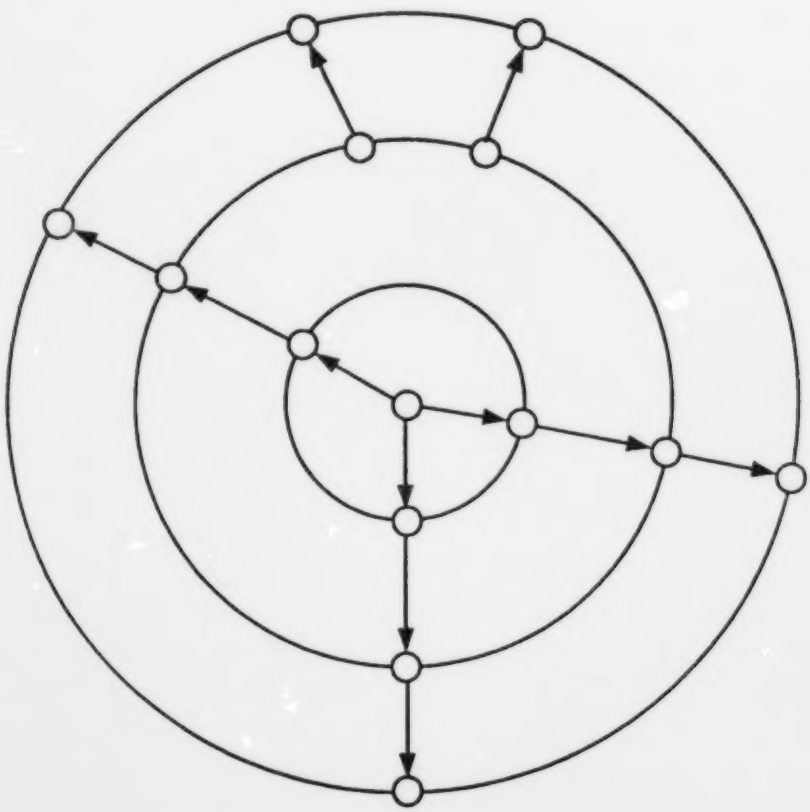


Figure 4: A general spiderweb network

It is interesting to point out that some well-known configurations can be generalized as spiderwebs; we have, for instance, a tree, a star, a ring, intersecting rings and even a simple line.

We now have all the elements to enunciate the following statement

Statement 1 *:The only possible cycles that can be formed in a spiderweb are those of lenght 2, and those that correspond to all the nodes of a given level.*

The above statement allows us to formulate the DPSCP for a spiderweb (DPSCP-sn). Let L represent the set of all levels of the spiderweb and F_k, the set of nodes in level k, then the following contraints should be added to the DPSCP formulation:

$$\sum_{i,j \in F_k} x_{ij}^t \leq |F_k| - 1 \qquad \forall t \in T; \quad \forall k = 1, \ldots |L|. \tag{7}$$

The DPSCP-sn can thus be formulated as follows:

(DPSCP-sn): Min (1) subject to (2), (3), (4), (6) and (7).

An illustrative spiderweb network example is depicted in Figure 5, results are tabulated in Table 4.

VARIABLES	NODES STASHED	STASHING TIME
x_{14}, x_{47}, x_{78}	4,8	t=2
$x_{12}, x_{23}, x_{36}, x_{69}$	6,9	t=3
x_{14}, x_{47}, x_{78}	4,8	t=4
$x_{14}, x_{47}, x_{69}, x_{76}, x_{78}$	4,6,8,9	t=6
$x_{14}, x_{47}, x_{69}, x_{76}, x_{78}$	4,6,8,9	t=8
COST OF THE LP RELAXATION	34	
COST OF THIS SOLUTION	48	

Table 4: **Results for the spiderweb network example**

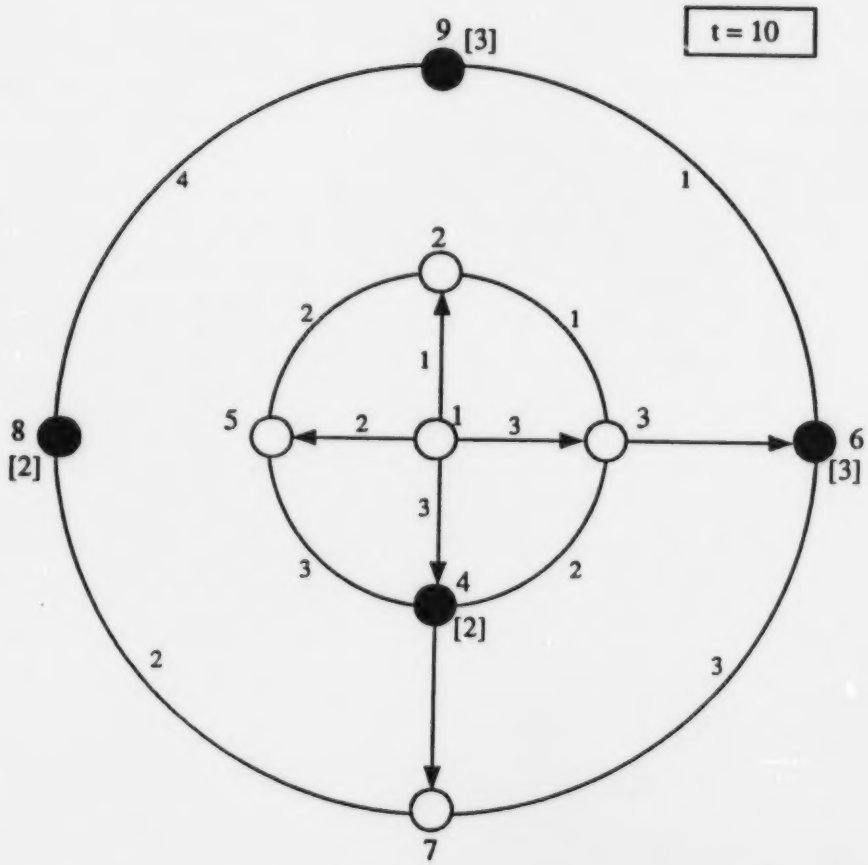


Figure 5: A spiderweb network example

6 Conclusions and further work

We have formally introduced the DPSCP as a mathematical programming problem. Several formulations were presented according to the network topology. The problem, for the general case, can be classified as NP-complete. However, it has been shown that the problem can be solved optimally for small networks. The mathematical programming approach can also be useful for medium-sized networks, in that case, the branching procedure could be stopped before reaching the optimum and bounds on how good the optimal solution is should be obtained. For larger networks, heuristics should be designed.

We have made the conjecture that the DPSCP solution presents a periodic behavior that depends on the predicate requirements and the inital state of the stashing. The study of such periodic behavior constitutes an interesting avenue of research to be able to solve the problem for large networks.

The DPSCP could also be studied under a stochastic optic. For instance, let us suppose the time predicates are relaxed x%of the time, in other words, that the user could demand the predicate to be respected only $(100-x)$% of the time. How would the problem be modeled for that case? Another stochastic aspect to be considered would be the reliability problem applied to the DPSCP. In fact, the whole idea of stashing a file comes from the desire of increasing its availability, specially in the event of failure. Thus, the probability of failure of nodes and links could greatly influence the optimal stashing strategy. We are planning to tackle this problem in the near future.

References

[1] Alonso R., D. Barbará and L. Cova. 1990. "Using stashing to increase node autonomy in distributed file systems". *Proceedings of the ninth symposium on reliable distributed systems.* Huntsville, Alabama, October 1990, pp 12-21, IEEE-CS press.

[2] Alonso R. and L. Cova. 1990. "Managing replicated copies in very large distributed systems". *Proceedings of the workshop on management of replicated data*, Houston, Texas, November 1990, IEEE-CS press.

[3] Balachandran V. and S.D. Deshmukh. 1976. "Storage decisions in information systems". *AIIE Transactions*, volume 8, No. 3, pp 358-364.

[4] Birrell A. 1988. Position paper of the 1988 ACM SIGOPS European Workshop, Cambridge, England, September, 1988.

[5] Bouliane J. 1988. "Minimisation du nombre de relais dans les tournees des facteurs". Master's thesis. Centre de recherche sur les transports, Université de Montréal, CRT-607.

[6] Bouliane J. and G. Laporte. 1989. "Locating postal relay boxes using a set covering algorithm". Centre de recherche sur les Transports, Université de Montréal, CRT-611.

[7] Cova L. 1990. "Resource management in federated computing environment". Dissertation. Department of Computer Science, Princeton University. Publication number CS-TR-282-90.

[8] Desrosiers J., F. Soumis and M. Desrochers. 1984. "Routing with time windows by column generation". *Networks 14*, pp 545-565.

[9] Eppstein D., G.F. Italiano, R. Tomassia, R.E. Tarjan, J. Westbrook, M. Yung. 1990. "Maintenance of a minimum spanning forest in a dynamic planar graph". Computer Science Department, Princeton University. CS-TR-243-90.

[10] Garey M.R. and D.S. Johnson. 1979. "Computers and intractability: A guide to the theory of NP-completeness". W.H. Freeman and company editions.

[11] Karlin A.R., M.S. Manasse, L. Rudolph, D. Sleator. 1988. "Competitive snoopy caching". *Algorithmica*. volume 3, pp 79-119.

[12] Kistler J.J. 1990. "Increasing file system availability". *Proceedings of the workshop on management of replicated data*, Houston, Texas, November 1990, IEEE-CS press.

[13] M.S. Manasse, L.A. McGeoch and D.D. Sleator. 1990. "Competitive algorithms for server problems". *Journal of Algorithms*, volume 11, pp 208-230.

[14] Mullender S. 1989. "Distributed computer systems". *Addison-Wesley*.

[15] Walker B., G. Popek, R. English, C. Kline, and G. Thiel. 1983. "The LOCUS distributed operating system". *Proceedings of the 9th ACM Symposium on Operating System Principles.*